RELATIONAL DATABASE ASSIGNMENT

Joseba Hernández Bravo

December 20

1 Implementation

Before we start explaining how we have created the data base, it is important to define all the values that will go into it. Since the choice of the theme was optional, I decided to create a database to store all the characteristics about the **climbs** made in the last years. Moreover, this database is designed to store not only my personal information, but also that of the rest of the users who are willing to collaborate.

This idea is inspired by the 8a.nu[1] website which stores information about the climbs (sport climbing routes and boulder problems) of the users from all over the world. However, it does not allow data on big wall climbs, which we will add in this work.

The relational data base will contain the following tables:

- **climber**: contains the information about the user/climber: name, surname, age and the routes that he/she climbed.

- **b_wall**: it contains a table of different big wall (as the walls from Yosemite, Patagonia, Vignemale...) climbing routes with the information about the difficulty[2] of the hardest pitch, information about the progression style (free-climbing, solo climbing, artificial climbing ...), information on whether or not the user free climbed the route, the country of the wall, the location of the wall and the route-setter.

[1] Link to the web page: https://www.8a.nu/

[2] We will use the French notation.

https://freeman.la/los-grados-la-escalada-roca-breve-explicacion-tabla-conversion/

- **setter**: it contains the information about the route-setter of the big wall routes.
- **country**: this table contains a information about the countries used in b_wall table.
- **sport_climbing**: as in the case of the big wall climbs, here we collect the information about the routes, but in this case, about the routes climbed in a sport-climbing areas as Margalef, Siurana, La Hermida ... . Here we store the information about the grade of the route, if the route was climbed onsight[3] or flashed[4] in case the route was climbed on the first try, number of attempts that have been necessary to achieve (from 0 to inf), some invented scores and the information about the location of the route (the community and the spot)
- **comunity**: contains information about the autonomous community to which the different routes belong.
- **spot**: contains information about the climbing spots or locations.

Once we know the elements of our database we can continue with the next step: ER-diagram. The first step is to create the relational diagram on paper. In my case, I have created it in my personal notebook. Once this is done, having in mind the shape we want to give to our database, we create a script (explained in the next section) and we run it using *dbeaver−ce*.

[3]Onsight climbing: https://www.climbernews.com/what-does-climbing-onsight-mean/
[4]Flash climbing: https://www.climbernews.com/what-is-a-flash-in-climbing/

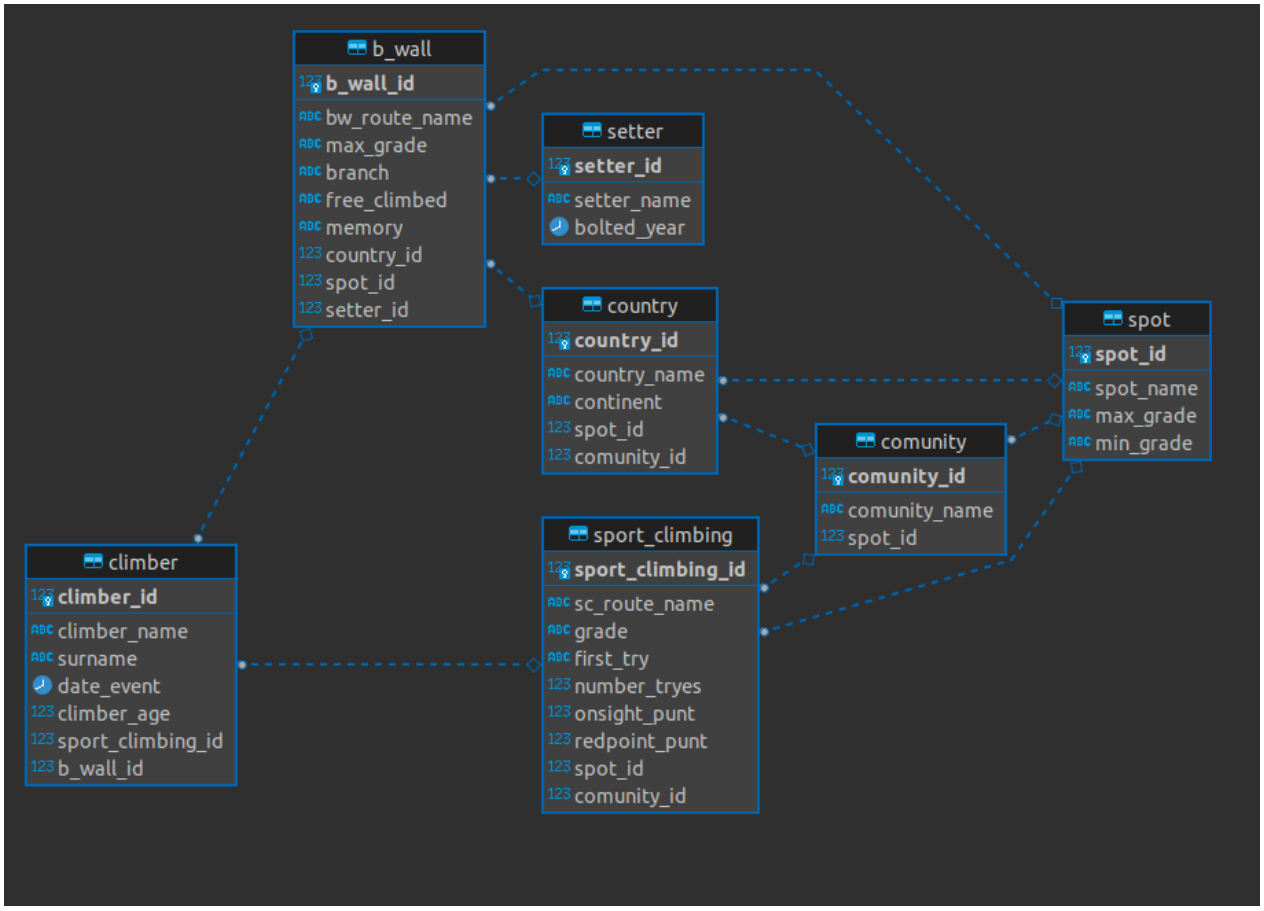


Figure 1: ER-Diagram of our relational database.

This model has some normalisation rules:

- **1.NF:** Each cell contain only a single (atomic) value, every column in the table is uniquely named and, All values in a column pertain to the same domain. For example for each user we have created a table with the different climbing routes.
- **2.NF: Create separate tables for value sets that apply to multiple records.** An example of this could be the **spot** table. 4 of our tables uses this information, so we decide to separate it in a new table.
- **3.NF** We have removed fields that do not depend on the key.

1.1 creation

Creating the database is very simple. First of all, make sure you have postgre[5] installed on your computer. Once you have it installed, simply create the database using the following commands:

```
sudo su postgres
```

```
createdb myClimbsdb
```

Then, we will connect to the database created using *dbeaver-ce*.

Finally we will a sql script[6], and we will execute it.

2 Queries

In this section we will do three examples of a creation queries tha involve at least two tables:

- Imagine that we want to know the sport climbing and b_wall names that are linked to all the users:

```
select  c.climber_name , c.surname , bw.bw_route_name , sc.sc_route_name
from climber c
full join b_wall bw
on c.b_wall_id  = bw.b_wall_id
full join sport_climbing sc
on c.sport_climbing_id = sc.sport_climbing_id
```

here we ere involving three tables:

[5]Instalation guide: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-20-04-es

[6]Link Code: https://github.com/joseb061/Master-projects/blob/master/distributedsystems/RelationalDB/Script.sql

select c.climber_name , c.surname , bw.bw_route_name , Enter a SQL expression to filter resu

Grilla / Texto

	climber name	surname	bw route name	sc route name
1	Joseba	Hernandez	[NULL]	Malatesta
2	Joseba	Hernandez	[NULL]	Serra de prades
3	Joseba	Hernandez	Muga beroa	[NULL]
4	Joseba	Hernandez	culoir Amy	[NULL]
5	Mikel	Aurrekoetxea	[NULL]	Transilvania
6	Mikel	Aurrekoetxea	[NULL]	Escamarlà

Figure 2: Route names per climber and type of route.

- Let's imagine now that we want to find the list of sport climbing routes and their respective grades by autonomous community and that we also want to order the lines according to the grade of the route (in this case from highest to lowest) :

```
select sc.sc_route_name ,c2.comunity_name, sc.grade , sc.redpoint_punt, sc.onsight_punt
from sport_climbing sc
full join comunity c2
on sc.comunity_id = c2.comunity_id
order by sc.redpoint_punt desc;
```

This will be the result:

	sc route name	comunity name	grade	redpoint punt	onsight punt
1	Malatesta	Basque country	8b	81,66	0
2	Transilvania	Catalunya	8a	80,66	0
3	Escamarlà	Catalunya	7c+	78,34	0
4	Serra de prades	Catalunya	7c	0	140

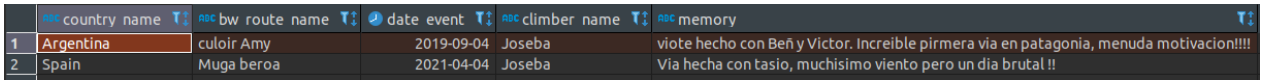
Figure 3: Different sport climbing routes ordered by respoint_ punt.

Finally, let's imagine that we want to see all the tracks made on the big wall, but we are only interested in seeing the ones made by a particular user.

```
select c.country_name , bw.bw_route_name,c2.date_event , c2.climber_name, bw.memory
from country c
inner join b_wall bw on bw.country_id = c.country_id
inner join climber c2 on c2.b_wall_id  = bw.b_wall_id
```

```
where c2.climber_name = 'Joseba'
order by c2.date_event asc;
```

and the result will be:



	country_name	bw_route_name	date_event	climber_name	memory
1	Argentina	culoir Amy	2019-09-04	Joseba	viote hecho con Beñ y Víctor. Increible pirmera via en patagonia, menuda motivacion!!!!
2	Spain	Muga beroa	2021-04-04	Joseba	Via hecha con tasio, muchisimo viento pero un dia brutal !!

Figure 4: Memories from the big wall routes ordered by the date.

3 Insert data and make a backup

If we want to insert new data to our database we only need to use the command INSERT:

```
insert into climber (climber_name, surname,date_event,climber_age,
                                        sport_climbing_id, b_wall_id)
values ('Tasio','Martin','14/10/2021',23,2,Null);
```

Here we are inserting a new climber called 'Tasio Martin', who has climbed the 2. climbing route.

Finally, we will make a backup of our database from the terminal:

```
pg_dump {h 127.0.0.1 {U <user1> database | gzip > database.dmp.gz
```

in my case:

```
pg_dump -h 127.0.0.1 -U postgres myClimbsdb | gzip > myClimb.dmp.gz
```